9系统调用

2019年2月8日 16:30

-
- 系统调用的概念和类型

1. 程序接口是OS给用户程序设置的取得OS服务唯一途径，由系统调用构成
2. 系统调用system call：既提供了用户程序和OS内核的接口，又可给OS自身使用，每个系统调用都是能完成一特定功能的子程序

一. 系统调用的基本概念

1. 系统态/核心态和用户态
   1) 特权指令：系统态运行的指令，只能OS使用。既能访问用户空间，也能访问系统空间。启动外部设备，设置时间，关中断，执行状态转换等
   2) 非特权指令：用户态运行的指令，为防止程序异常破坏系统，不可直接访问系统硬件和软件，只能完成一般性的操作和任务
   3) 程序使用特权指令会发出权限错ixnh，系统转入错误处理程序，停止运行该程序，重新调度
2. 系统调用与一般过程调用的区别
   1) 主调程序在用户态，被调程序却在系统态
   2) 需要先通过软中断机制切换到系统态，再经内核分析，才能转向调用
   3) 抢占式调度系统中，调用的过程执行完后可能根据优先级执行其他进程
   4) 嵌套调用一般有深度限制，如有的系统最大深度为6
      (1) 例：拷贝文件，再指定文件名后，需要依次调用
      open、creat、alloc、read、close、write、close，任一调用出错都需要调用exit正常结束程序
3. 中断机制。如MS-DOS的INT21H
   1) 系统调用都是通过中断机制实现的，每个系统调用都通过中断入口实现
   2) 应当是被保护的，如IBM PC上Intel提供了多达255个中断号，未授权给应用程序保护等级的中断号被应用程序调用后会引起保护异常，导致被终止；Linux则只给了3，4，5，80h四种中断号，第四个是系统调用中断号

二. 系统调用的类型

1. 进程控制类：创建和终止进程、获得和设置进程属性、等待某事件等
2. 文件操作类：创建删除、打开关闭、读写等
3. 进程通信类：基于连接的消息传递、基于虚地址空间共享存储区的通信
4. 设备管理类：申请释放、设备IO、重定向、获得设置属性等
5. 信息维护类：获得时间、获得操作系统版本、获得当前用户、获得空间大小等

三. POSIX标准Portable Operating System IX：基于UNIX的可移植操作系统接口

1. 定义了标准API、保证程序源代码可兼容多系统移植运行
2. 定义了一组构造操作系统必须的过程，大多数系统调用应对应一个或一组过程
3. 并没有指定系统调用的实现形式，早期流行汇编，新推出的系统中常用C语言写系统调用，以库函数形式提供，隐藏了访管指令的细节

4.

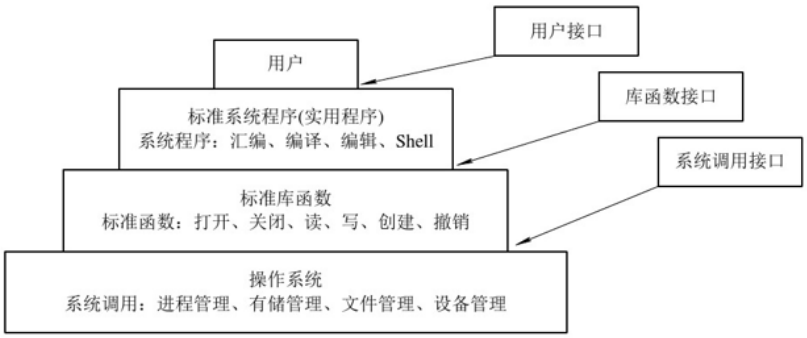

图9-7 UNIX/Linux系统程序、库函数、系统调用的分层关系

-
- UNIX系统调用

一. 进程控制

1. 进程的创建和终止
   1) 创建子进程fork，继承调用者进程的各种环境、文件、根目录、当前目录、大多数属性，进程映像也基本相同
   2) 终止进程exit，创建子进程时一般在末尾安排一exit，使其自我终止，留下一条记账信息status
2. 改变进程映像和等待
   1) 执行文件exec，覆盖调用者的进程映像
   2) 等待子进程结束wait，将调用者挂起，终止与子进程的同步
3. 其他进程调用
   1) 获得进程ID，如getp-id活动标识符、getpgrp活动进程组id，getppid获得父进程id
   2) 获得用户id，如getuid获得真正id，geteuid获得有效id，getgid获得真正用户组id
   3) 进程暂停pause，挂起进程直至收到信号

二. 文件操纵

1. 文件的创建和删除
   1) 创建或重写同名文件creat，并打开，返回其描述符fd，失败则返-1
   2) 删除文件在UNIX无法通过对应系统调用实现，无连接时才能删
2. 文件的打开和关闭
   1) 打开open将文件从硬盘拷到内存，返回描述符fd，作用是代替路径名
   2) 关闭close断开程序与文件的快捷通路，访问计数为0后才能真正关闭
3. 文件的读和写
   1) read和write都需要提供三个参数：描述符fd、缓冲区首址buf（读的目标地址或写的源地址）、需传送字节数nbyte
4. 建立与文件的连接和去连接
   1) 连接link，实现文件共享，并给连接用户进程数+1
   2) 去连接unlink，实现断开连接，并给连接计数-1，为0后才能删除

三. 进程通信和信息保护

1) 软件包IPC专门用于实现进程通信，包括消息机制、共享存储区机制、信号量机制，每个机制中都提供了对应的系统调用

1. 进程通信
    1) 消息机制：msgget建立信息队列，成功则返回消息队列描述符msgid用于访问该消息队列；msgsend用于发生消息给队列；msgrcv用于从指定队列接收指定类型的消息
    2) 共享存储区机制：先用shmget建立共享存储区，成功则返回共享存储区描述符sgmid；shmat将该共享区连接到进程的虚地址空间上；shmdt拆除进程与共享存储区间的连接
    3) 信号量机制，同二章，可将一组信号量形成一个信号量集，执行原子操作
2. 信息维护
    1) 设置时间stime，获得时间time
    2) 获得进程和子进程使用CPU时间times，包括在用户空间执行指令时间、调用进程时间、子进程在用户空间使用CPU时间、系统为各子进程花费CPU时间，这些时间可填写到指定缓冲区
    3) 设置文件访问和修改时间utime，如果参数times为NULL，有写权限的以后可修改为当前时间，否则times为执行utim buf结构的指针，以后只能讲访问时间和修改时间置入该结构中
    4) 获得UNIX系统名称uname，相关信息存在utsname结构中，包括系统名字符串、系统在网络中的名称、硬件的标准名称等

-
- 系统调用的实现

一. 系统调用的实现方法

1. 系统调用号和参数的设置
    1) 一般每条系统调用对应唯一系统调用号
    2) 在系统调用命令（陷入指令）传递调用号给中断和陷入机制的方法
        (1) 直接放在系统调用命令（陷入指令）中，如IBM370和早期UNIX
        (2) 装在寄存器中，如MS-DOS放AH，Linux放EAX
    3) 将系统调用所需参数传给陷入处理机构和系统内子程序/过程的方法
        (1) 让陷入指令自带少量有限参数
        (2) 用相应寄存器，传递有限数量的参数，如MS-DOS用MOV指令
        (3) 参数表方式，只将参数表指针存进寄存器，如UNIX和Linux
2. 系统调用的处理步骤
    1) 设置完调用号和参数后，UNIX会执行CHMK命令，MS-DOS会执行INT-21软中断
    2) 首先处理状态由用户态转为系统态，由硬件和内核程序进行系统调用的一般性处理，主要是转移上下文到堆栈和保存参数到指定地址
    3) 之后根据调用号查系统调用入口表，以转入对应子程序
    4) 最后恢复CPU现场，继续往下执行
3. 系统调用处理子程序的处理过程

1) 系统调用的功能主要是由系统调用子程序来完成的
2) 如Creat命令的子程序中，核心会根据文件路径名查找指定目录，如果已存在无写权限的文件，会认为出错；如果已存在有权限的文件，就释放该盘块，准备写入新数据文件；如果无指名文件，则表示要创建一个新文件，核心需要找出一个空目录项，初始化，再打开新建文件

二. UNIX系统调用的实现

1) UNIX系统V的内核中有一个trap.S文件，它是中断和陷入总控程序，用于一般性处理中断和陷入。为了效率，由汇编语言编写
2) 还有一个C语言编写的trap.C程序，专门处理系统调用、进程调度中断、跟踪自陷非法指令、访问违章、算术自陷等12种陷入的公共问题。主要包括确定系统调用号、传送参数、转入相应子程序等

1. CPU环境保护
   1) 用户态，执行CHMK命令前，应填好参数表，将地址传进R0寄存器
   2) 执行CHMK命令后处理机转为核心态，硬件自动将处理机状态字长PSL、程序计数器PC、代码操作数code等压入以后核心栈，再转入trap.S
   3) trap.S执行后，将陷入类型type和用户栈指针usp压入用户核心栈，再通过特定寄存器的屏蔽码，把对应寄存器中的CPU环境也压入栈
2. AP和FP指针
   1) AP指向参数表，FP指向调用栈帧，指示本次系统调用所保存的数据项
   2) 出现新系统调用时，需将AP和FP303压入栈，实现了嵌套系统调用
   3) trap.S完成了CPU环境和AP、FP的保存后，调用trap.C完成后续
3. 确定系统调用号
   1) trap.C的调用形式为trap(usp,type,code,PC,PSL)
   2) 令i=code&0377。0<i<64时i即为系统调用号，i为0时需通过间接指针
4. 参数传送
   1) 指由trap.C将系统调用参数表的数据从用户区传到User结构的U.U-arg中
   2) 根据系统调用定义表规定的参数个数进行传送，最多10个
5. 利用系统调用定义表Sysent转入相应的处理程序
   1) 该表是个结构数组，每个结构里有所需参数数、待传参数数、子程序入口
6. 系统调用返回前的公共处理
   1) UNIX进程动态优先级随时间加长而降低，每次系统调用返回trap.C都需重新计算优先级。另外，系统调用执行时发生错误时会设置再调度标志，处理子程序在计算优先级后若检查到该标志，便会调用switch调度程序，选择就绪队列的最高优先级进程，转交处理机给它运行
   2) 当进程处于系统态时，不理睬其他进程发来的信号；回到用户态时，内核才坚持该进程是否有收到信号并执行相应动作。处理结束后执行返回指令RET，将已被压入用户核心栈的数据退还给相应寄存器、让进程继续执行

三. Linux系统调用

1. 和UNIX相似。最多有190个系统调用
2. Linux在CPU的保护模式下提供了四个特权级别，目前内核都只用到两个：0级

内核态和3级用户态

3. 用户对系统调用不能任意拦截和修改，以保证内核安全
4. 每个系统调用的组成部分
    1) 内核函数，作为核心驻留在内存，由C书写，运行在内核态，一般不能调用其他系统调用或应用程序可用的库函数
    2) 接口函数，是提供给应用程序的API，以库函数形式存在lib.a中，由汇编语言书写，主要功能是把系统调用号、入口参数地址传给相应核心函数，并使用户态下运行的应用程序陷入核心态
5. 由汇编写的系统调用入口程序entry(sys_call_table)
    1) 包含了系统调用入口地址表，给出了系统调用核心函数名
    2) 每个核心函数的编号定义在include/asm/unistd.h：

        ENTRY(sys-call-table)

            long SYMBOL_NAME(sys_xxx)i
    3) Linux的系统调用号即系统调用入口表中位置序号
    4) 所有系统调用通过接口函数将调用号传给内核，内核转入系统调用控制程序，通过调用号定位核心函数，Linux内核陷入由0x80(int80h)中断实现
6. 系统调用控制程序的工作流程：取系统调用号，检验合法性；执行int80h产生中断；转换地址、切换堆栈、转内核态；中断处理，通过调用号定位内核函数；通过寄存器内容从用户栈取入口参数；执行核心函数，结果返回给程序

四. Win32的应用程序接口

1. 应用程序接口API是一个函数的定义，说明如何获得一个给定的服务；而系统调用是通过中断向内核发出的一个请求。API可能调用也可能不调用系统调用
2. Windows程序设计模式是事件驱动方式，与UNIX和Linux有根本的不同，主程序需要等待事件的发生，根据事件的内容，调用相应的程序
3. 通过调用Win32API可以创建文件、进程、线程、管道等对象，并将聚彬返回给调用者，调用者课根据句柄间接知道对象在内存的具体位置
4. Windows中，只有对操作系统性能起关键作用的程序才能运行在核心态，如对象与安全管理器、线程与进程管理器、虚存管理器、高速缓存管理器、文件系统等，它们构成了操作系统执行体executive
5. 在Intel x86处理机上，当程序调用操作系统服务时，需要执行int2E指令，由硬件产生陷入信号，系统捕捉后切换到核心态，控制权转交给陷入处理程序的系统服务调度程序，该程序负责关中断、保存现场、检查参数、并转移到核心态堆栈、查找系统服务调度表/陷入向量表以获得对应服务的地址并转交控制权
6. 支持API的三大组件Kernel、User、GUI

    (1) Windows将这三个组件置于DLL动态链接库dynamic link library中

    (2) 任何应用程序都共享这三个模块的代码，可直接调用其函数

    1) Kernel包含了大量操作系统函数，如内存、进程的管理等
    2) User集中了窗口管理函数，包括创建、撤销、移动、对话等
    3) GUI提供画图、打印等函数

        i.

ii.
iii.
iv.
v.
vi.
vii.
viii. ----------我是底线-------